Remarks

Summary: By this Amendment and Response, the specification has been amended in response to Paragraph 4; Claims 2-6, 10-14, and 16 have been canceled; claims 1, 7-9, 15, and 17-20 have been amended; and arguments are presented for the patentability of amended claims 1, 7-9, 15, and 17-20.

Specification Amendments: The paragraphs noted as 0001 and 0070 (starting at respective page 1, line 12; and page 25, line 8) have been amended to insert the Application Number. As to the paragraph noted as 0023, on page 4, the paragraph starting at line 2 has been amended to spell out DPF. As to the paragraph noted as 0058, counsel has been unable to locate the noted misspelling of "system 114ais", and notes that the suggested "system 14a is" has no antecedent. A search was made for "system 114ais" and did not locate that term. It is respectfully requested that the page and line of the asserted misspelled word be identified. In this regard, it is noted that at page 18, line 24, "system 114a is" appears; at page 20, first paragraph, last two lines, the next-to-last line ends with "114a" and the last line starts with "is"; and at page 21, lines 19 and 20, "system 114a is" appears. The amended title now conforms to the scope of the claims.

Response to Paragraph 9: Consideration of amended claims 1, 7-9, 15, and 17-20 as being patentable over Chung is respectfully requested. These claims now define more specific aspects of the distributed processing framework used to test software programs under the control of a software test application program. This testing presents unique problems in the distributed processing framework in terms of efficient use of the framework.

For example, amended claim 1 defines a test suite configured to define tests for testing each particular software program to be tested. Each test suite is configured to define a plurality of test execution requests for testing the particular software program. A plurality of software test systems is configured in the distributed processing framework, with each of the software test systems being configured to operate under control of a test application program to test a particular software program with respect to ability to perform specific operations. The application program is configured to execute one of the test suites that defines the tests to be

performed on the particular software program. First and second test execution requests for testing the particular software program are configured to be executed on different ones of the software test systems distributed among the distributed processing framework.

A system controller is provided for locating a first software test system capable of executing the first test execution request. The system controller further locates a second software test system capable of executing the second test execution request. The respective located software test system executes a respective test execution request configured to monitor progress in the execution of the respective test execution request on the respective located software test system. The monitoring comprises gathering data identifying each software test system that is executing at least one of the test execution requests and periodically determining a point of the execution of the respective test execution request;

A respective one of the located software test systems is capable of utilizing the identification of the software test system that is executing the at least one of the test execution requests of the gathered data to identify a point of the execution of a respective test execution request being executed by that one located software test system. In this manner, upon interruption of the execution of that at least one test execution request by that respective one located software test system, the point of execution of that at least one test execution request by the one software test system is known. That respective one located software test system is effective upon such interruption to use the gathered data corresponding to that interrupted execution request to reinitialize the test application program corresponding to the respective one located software test system. At reinitializing, execution of the respective interrupted test execution request begins from a position described by the corresponding point of execution information.

Amended claim 9 defines method aspects for restoring execution of a test of a software program after interruption of the test performed in a distributed processing framework by a software test application program. The method provides a plurality of software test systems in a distributed processing framework. The software test systems are configured to test a particular software program with respect to ability to

perform specific operations. The software test application program is configured to execute a test suite that defines the tests to be performed on a particular software program. Importantly, the test suite (for one of the particular software programs to be tested) is divided into a plurality of test execution requests. First and second test execution requests are provided for testing one particular software program to be executed on different ones of the software test systems, wherein those systems are distributed among the distributed processing framework.

Progress in executing each test execution request on the respective software test system is monitored. The monitoring comprises gathering data identifying each software test system that is executing at least one of the test execution requests and periodically determining a point of the execution of the respective test execution request. Upon interruption of the execution of one of the test execution requests, the gathered data is used to identify the software test system that was executing the interrupted test execution request. The identified software test system that was executing the interrupted test execution request is identified. The reinitializing is at the determined point of the execution of the respective test execution request that corresponds to the identified software test system.

Also, amended claims 17-20 define a computer program embodied on a computer readable medium for restoring execution of testing of a software program after interruption of the testing. The testing is performed by various ones of a plurality of software test systems, each of the software test systems being configured to perform one or more tests of a test suite of the testing and being arranged in a distributed processing framework to execute a software test application program. The computer program comprises a first code segment that receives execution information from the software test application program. The execution information includes an identification of a respective software test application program being executed, an identification of the one software test system that is executing the respective software test application program with respect to one of the tests, and a current point of execution within the test suite. Another code segment updates a post mortem object based on the execution information. The update is specific to the respective software test application program being executed by the one software test system that is executing the one of the tests of the test suite, and is specific to the current point of

execution within the test suite.

A further code segment reinitializes that respective software test application program on the respective one software test system utilizing the updated post mortem object after interruption of the one test executed by the one software test system that is executing the respective software test application program.

In the manner of the amended claims, access may be had to many distributed software test systems, and to ones capable of executing the particular test execution requests. Moreover, each such test system is able to gather information necessary to initialize execution of the respective interrupted test execution request, which begins from a position described by the corresponding point of execution information.

Consideration of amended claims 1, 7-9, 15, and 17-20 have as being patentable over Chung is respectfully requested for the following reasons. Chung is characterized by a problem in use of general software, and not in the testing of software under the control of software test application programs enabled to run on a distributed processing framework. Initially, Chung focuses on the persistent state in running the general software (C2, L15-21, C2, L45-47, for example). Chung refers to a user application process (C2, L63) which is executed, as understood, on one machine and not based on use of a software test suite that is divided into separate tests to be run on separate test systems.

As to the focused-on persistent state in relation to checkpointing, (C3, L1-3, C3, L28-30, and C12, L49), there is no recognition of the claimed aspect of software testing in which a test suite is divided into many separate tests, i.e., in the claimed manner of the individual software tests, and no checkpoint ability corresponding to each such test.

The many processing units of Chung are not the claimed specific test systems for test execution requests. For example, the export noted at C5, L10-12 is not related to a test execution request, and instead relates to recovery at the remote node when recovery at the local node is not successful.

As to the claimed monitoring, Chung does not teach or suggest dividing a test suite into separate tests, nor monitoring each test in terms of a related test execution request (claims 1 and 9). At C6, L19 as to each of Chung's application processes, there is no monitoring of individual tests on different test machines configured to run

the individual tests. Instead, each entire application process is monitored by monitor 85. Further, as to the Chung monitoring in active or passive manners, the whole Chung application process is monitored, which neither teaches nor suggests tests run on different test systems under the same software test application system, as claimed (C6, L23+ and 31+ and L65). Further, the Chung monitoring is based on future altering via monitoring file system calls (C7, L55) and intercepting those calls with respect to each entire UAP.

As to executing the user application program, at C9, L46+ the entire UAP is of concern, not the claimed test-by-test processing under a software test application running at many test systems. Thus, Chung does not suggest individual test execution requests that are separately processed on separate software test systems.

As to saved data for checkpoint purposes relating to one UAP, at C10, L64-67 the data relates to the entire UAP and not as claimed to running one of many tests on a particular software program, where the tests may be executed at the many test systems.

As to amended claims 17-20, the computer program is embodied on a computer readable medium for restoring execution of testing of a software program after interruption of the testing. The testing is performed by various ones of a plurality of software test systems, each of the software test systems being configured to perform one or more tests of a test suite of the testing. Each such system is arranged in a distributed processing framework to execute a software test application program. As to the claimed code segment for receiving, Chung does not teach or suggest a code segment that receives information about a test defined as part of a test suite divided into separate tests. In detail, the claimed computer program comprises a first code segment that receives execution information from the software test application program. The execution information includes an identification of a respective software test application program being executed, an identification of the one software test system that is executing the respective software test application program with respect to one of the tests, and a current point of execution within the test suite. On the other hand, the code segment in Chung receives data based on future altering via monitoring file system calls (C7, L55) and intercepting those calls with respect to each entire UAP. Thus, there is no suggestion of a code segment

receiving execution information that includes an identification of one software test system that is executing a software test application program with respect to one of many tests defined by a test suite, nor receiving a current point of execution within such a test suite.

As to the other code segment, Chung does not have a code segment that updates a post mortem object, where the update is specific to a respective software test application program that is being executed by the one software test system that is executing the one of the tests of a test suite. As a result, in Chung there is no update specific to the current point of execution within such a test suite.

As to the further code segment, Chung does not have a code segment that reinitializes that respective software test application program on the respective one software test system utilizing the updated post mortem object after interruption of the one test executed by the one software test system that is executing the respective software test application program, where the one test is of many defined by a test suite.

In view of these remarks, consideration of the pending claims as being patentable over Chung is respectfully requested, and allowance of this application is believed to be in order, and is respectfully requested.

Should the Examiner have any questions concerning this Application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,

MARTINE PENILLA & GENCARELLA, LLP

Chester E. Martine, Jr.
Reg. No. 19,711

710 Lakeway Drive, Suite 200
Sunnyvale, CA 94085
Telephone (408) 774-6908
Customer No. 32291